

TP 4 : Segmentation

Nous avons vu dans le TP 3 une première approche de la segmentation : la segmentation binaire, c'est-à-dire la division des pixels de l'image en une classe "fond" et une classe "objets". Lorsque l'on détecte plusieurs objets, il peut être intéressant de traiter séparément ces objets, il faut donc pouvoir les identifier dans la classe "objet". Un certain nombre de fonctions Matlab permettent de rechercher et d'indexer les ensemble de pixels connectés dans une image binaire. Le but de ce TP est de se familiariser avec ces fonctions et d'améliorer la qualité de la segmentation et de la détection d'objets en prétraitant l'image à segmenter.

1 Segmentation par seuillage binaire

L'image 'rice.png', présente dans la librairie d'images de Matlab, est une photographie de grains de riz, le but de cette section est d'effectuer une segmentation de cette image pour détecter séparément les différents grains de riz.

Manipulation 1

- ➡ Charger et afficher l'image 'rice.png' présente dans la librairie d'image Matlab.
 - ➡ Effectuer le seuillage automatique de cette image (on pourra réutiliser le code du TP3).
-

A partir du résultat de la segmentation binaire, on souhaiterait segmenter séparément les grains de riz. Pour cela, nous allons utiliser les fonctions codées dans Matlab pour identifier séparément chaque ensemble de pixels connectés et visualiser le résultat.

```
>> CC = bwconncomp(im_seuil); # Identifie les ensembles de pixels connectés
                                dans l'image binaire im_seuil.
>> L = labelmatrix(CC);        # Crée une matrice L de la taille de l'image
                                qui va contenir les labels de chaque ensemble
                                de pixels connectés identifié dans CC.
>> im_couleur = label2rgb(L); # Crée une image RGB où chaque label est
                                associé à une couleur.
```

Manipulation 2

- ➡ Appliquer ces fonctions à l'image binaire obtenue précédemment.
-

🔗 Commenter le résultat obtenu.

2 Correction du défaut d'éclairage

L'image que l'on souhaite segmenter présente un défaut d'éclairage qu'on n'a pas mesuré. On souhaite estimer ce défaut afin de corriger au mieux l'image avant de la segmenter (et éviter ainsi les artefacts détectés sur la segmentation précédente).

2.1 Estimation du défaut d'éclairage et correction

Manipulation 3

- ➡ Charger et afficher l'image 'rice.png'.
 - ➡ Afficher son histogramme.
 - ➡ Afficher sur le même graphe la 10^{ème} colonne, la 50^{ème} colonne et la 120^{ème} colonne de l'image 'rice.png' (commande `hold on` à utiliser avant la commande `plot()`).
-

⚡ Que peut-on dire du défaut d'éclairage ?

⚡ Etant donné les observations faites précédemment, proposer une méthode pour estimer le défaut d'éclairage.

On rappelle quelques fonctions utiles pour faire des opérations sur les images en Matlab.

```
>> sum(I(:)); # Somme de tous les pixels de l'image.  
>> sum(I, 1); # Somme des lignes de l'image.  
>> sum(I, 2); # Somme des colonnes de l'image.
```

A noter que cette syntaxe est valable pour les fonctions `mean` et `median` également.

Manipulation 4

- ➡ Appliquer la correction d'éclairage.
-

⚡ Commenter le résultat obtenu après la correction d'éclairage. En quoi cela permettra d'obtenir une segmentation binaire de meilleure qualité ?

2.2 Segmentation

On souhaite maintenant étudier l'influence de la correction d'éclairage sur la segmentation.

Manipulation 5

- ➡ Appliquer la segmentation codée dans la partie 1 sur l'image corrigée.
-

⚡ Commenter le résultat obtenu.

3 Comparaison des deux méthodes

On fournit dans le fichier 'ground_truth.mat' la vérité terrain correspondant à la meilleure segmentation possible de l'image 'rice.png'.

Manipulation 6

- ➡ Comparer les performances des deux segmentations (avec et sans correction d'éclairage).
-

Différents critères ont été évoqués à la fin du cours sur la segmentation et pourront être utilisés pour évaluer les performances des deux segmentations.

⚡ Commenter les résultats obtenus.